СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭМУЛЯЦИОННЫХ ПРОЦЕССОВ НА ЛОГИЧЕСКИХ УРОВНЯХ АРХИТЕКТУРЫ УПРАВЛЯЮЩИХ КОМПЬЮТЕРОВ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ

COMPARATIVE ANALYSIS OF EMULATION PROCESSES AT LOGIC LEVELS OF ARCHITECTURE OF CONTROL COMPUTERS OF AUTOMATED CONTROL SYSTEMS

V. Zarubskiy

Summary. The basis of modern automated control systems are control computers with the property of functional redundancy at all logical levels of their architecture. The reliability of the operation of such control systems directly depends on the reliability of the operation of control computers. An increase in their reliability is seen in the practical implementation of the theory of structural stability, based on the implementation of emulation processes due to the properties of functional redundancy present in them. The article presents the results of an analysis of the possibility of implementing emulation processes at all logical levels of the architecture of control computers and draws conclusions about their effectiveness at various levels.

Keywords: reliability, control computer, structural stability, functional redundancy, emulation.

Зарубский Владимир Георгиевич
К.т.н., доцент, Пермский институт ФСИН России
volen3030@rambler.ru

Аннотация. Основой современных автоматизированных систем управления являются управляющие компьютеры, обладающие свойством функциональной избыточности на всех логических уровнях их архитектуры. Надежность функционирования таких систем управления напрямую зависит от надежности функционирования управляющих компьютеров. Повышение их надежности видится в практической реализации положений теории структурной устойчивости, основанной на выполнении эмуляционных процессов за счет присутствующих в них свойств функциональной избыточности. В статье представлены результаты анализа возможности реализации эмуляционных процессов на всех логических уровнях архитектуры управляющих компьютеров и сделаны выводы об их эффективности на различных уровнях.

Ключевые слова: надежность, управляющий компьютер, структурная устойчивость, функциональная избыточность, эмуляция.

Введение

Проведенные ранее исследования в области выявления функциональной избыточности логических уровней архитектуры ЭВМ [1, 2, 5–8] и ее использования в целях повышения надежности компьютерного сопровождения процессов управления различными системами повышенной ответственности (ракетно-космические системы, системы атомной энергетики, транспортные системы, системы вооружения и др.), свидетельствуют о значительных ресурсах живучести управляющих компьютеров (УК) специального назначения. Однако, реализация обнаруженных запасов структурной устойчивости УК невозможна без разработки эффективных алгоритмов адаптации УК как функциональной системы к стохастически меняющемуся функциональному состоянию (ф-состоянию).

Материал и методы исследования

Естественно, процессы адаптации на различных уровнях архитектуры УК (рис. 1) имеют отличительные стороны вследствие специфики организуемых на них эмуляционных процессов. Но в них должно быть и много общего, связанного с единой методологией структурной устойчивости [8], что позволяет утверждать о целесообразности построения некоторой базовой методики разработки алгоритмов адаптации, которая с небольшими изменениями могла бы быть использована в более широком смысле. Для обоснованного выбора базового уровня архитектуры настоящего исследования, очевидно, следует провести сравнительный анализ эмуляционных процессов на ряде логических уровней архитектуры. В качестве универсального эталона можно предложить характеристики готовности исправного УК, относительно которых предполагается подвергнуть сравнительному анализу все изменения, сопровождающие этапы возникновения и алгоритмической компенсации отказов аппаратуры, и вытекающие из этого задачи адаптации.

Исправный УК характеризуется тройкой $(\theta,\ T,\ A_0)$, где $\theta = \{\vartheta_i : |\theta|\}$ — система команд, $T = \{t_i : |T| = |\theta|\}$ — временные характеристики ее быстродействия, причем, $\theta \leftrightarrow T$, т.е. введенные множества находятся во взаимо-

Рис. 1. Обобщенная архитектура перспективных УК

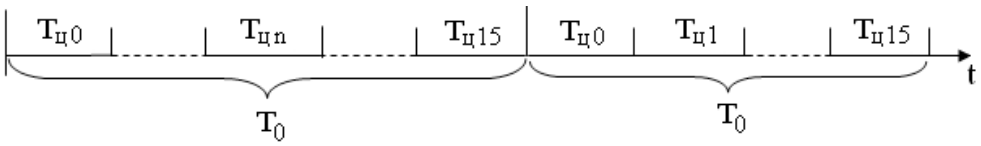

Рис. 2. Структура циклов работы УК

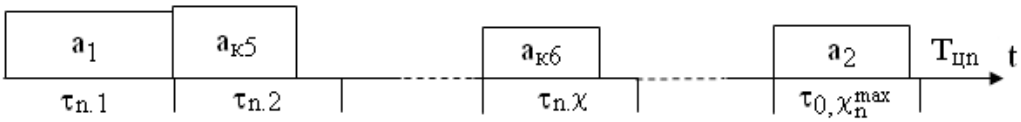

Рис. 3. Загрузка УК частными задачами управления из А

однозначном соответствии, $A_0 = \{a; |A_0|\}$ — множество алгоритмов (программ) специального программного обеспечения (СПО), на данном этапе исследования служащие временной характеристикой производительности, поскольку для каждого алгоритма а ∈ А его самая продолжительная реализация (доминанта) со смесью команд

$$\Xi_a^m = \left\{\xi_a^m; |\theta|\right\}$$

удовлетворяет отношению

$$\sum_{i=1}^{|\theta|} \xi_{ai}^m t_i \le \tau_a, a \in A_0, \tau_a \in \Im_0, A_0 \leftrightarrow \Im_0, \quad (1)$$

где τа — время, предоставляемое для решения а-го алгоритма в цикле УК Тц из расчета

$$\sum_{\chi=1}^{\chi_n^{max}} \tau_{n\chi} \le T_ц, n = \overline{0,15}, \tau_{n\chi} \in \Im_0, \quad (2)$$

что соответствует предельной загрузке УК на протяжении 16 циклов Тц, составляющих большой цикл УК Т0, охватывающий все задачи управления из А (рис. 2, 3).

На рисунке 3 показана типовая загрузка управляющих алгоритмов в цикле Тц УК согласно выражения (2). Традиционно в каждом цикле первые и последние алгоритмы повторяются: алгоритм а1 — является модулем выдачи готовности, а алгоритм а2 — модуль телеметрии. Остальные управляющие программы из множества А, $a_{к1}, a_{к2}, \ldots, a_{к8} \in$ А распределяются по телу цикла в соответствии с выполняемым этапом заданного режима системы управления.

Отказы в аппаратуре УК могут исказить ее логическое представление на одном или нескольких уровнях архитектуры в форме сокращения функциональной системы этих уровней. При этом УК теряет свою работоспособность даже в тех случаях, когда поврежденные

системы сохраняют функциональную полноту и потенциально способны решать все алгоритмы из А. Это обстоятельство объясняется тем, что микропрограммные и программные средства изначально разрабатываются из расчета целостности всех функциональных подсистем УК. Поэтому каждая утрачиваемая функция системы становится для этих средств нетерминальной, а сами средства — неприемлемы. Создание же альтернативных вариантов алгоритмов для каждого теоретически возможного функционально полного состояния УК нельзя считать перспективным из-за невозможности их размещения в ограниченной по емкости памяти.

Единственным способом поддержания работоспособности УК в условиях стохастически возникающих функциональных отказов (ф-отказов) следует считать восстановление утраченных функций на сохранившемся функциональном базисе, т.е. эмуляцию. Естественно, эмуляционные процессы влияют на характеристики восстанавливаемого УК, переводя тройку (θ, T, A) в тройку (θ', T', A') по схеме

$$(\theta,\ T,\ A) \rightarrow \text{ф-диагностирование} \rightarrow$$

$$\rightarrow \overbrace{\text{ф} - \text{адаптация} \rightarrow \text{А} - \text{аддаптация}}^{\text{Адаптация}} \rightarrow (\theta',\ T',\ A'), \quad (3)$$

функциональное диагностирование (ф-диагностирование) — логическое представление УК после проявления отказа в аппаратуре [3], функциональная адаптация (ф-адаптация) — восстановление работоспособного состояния УК [4], алгоритмическая адаптация (А-адаптация) — установление состава алгоритмов, удовлетворяющих требованиям отношения (1).

Рассмотрим особенности предложенной схемы адаптации на различных уровнях архитектуры.

Логический уровень аппаратного базиса архитектуры УК

В нерезервированной аппаратуре УК отказы физических элементов электронных схем немедленно приводят к распространению отказов на логические уровни согласно объективно существующим причинно-следственным связям, переводя УК в класс неработоспособных:

$$B_1 \rightarrow \Phi_1 \rightarrow M_1 \rightarrow (\theta_1,\ T_1,\ -), \quad (4)$$

где $\Phi_1 \subset \Phi$ — исправные физические элементы, $\Phi_1 \subset \Phi$, $M_1 \subset M$, $\theta_1 \subset \theta$ –функциональные состояния аппаратного базиса, микропрограммного и командного уровней архитектуры, неспособные обеспечить решение алгоритмов из A_0, предусматривающих наличие полной системы команд.

Если отказ УК компенсируется эмуляционными процессами на логическом уровне аппаратного базиса [6, 7] перебором ограниченного числа вариантов настройки (ф-диагностирование + ф-адаптация) толерантных элементов, то система микрокоманд как функциональная система восстанавливается, частично потеряв быстродействие относительно восстановленных микрокоманд M\M1:

$$t'_{\mu_j} > t_{\mu_j},\ \mu_j \in M \setminus M_1. \quad (5)$$

Тогда неизбежным станет снижение быстродействия команд из числа $\theta\backslash\theta 1$

$$t'_i > t_i,\ \theta_i \in \theta \setminus \theta_1, \quad (6)$$

что следует из

$$t'_i = \sum_{j_u=1}^{|M|} \xi_{Mj_u} t_{Mj_u} + \sum_{j_u=|M_1|+1}^{|M\setminus M_1|} \xi_{Mj_u} t'_{Mj_u} \quad (7)$$

и (5). Это приводит к необходимости поиска наилучшего варианта управления подконтрольным объектом в ранжированном ряде A = {A_0, A_1, …, A_η, …, $A_{\eta_{max}}$ } удовлетворяющего отношению

$$\sum_{\nu=1}^{|\theta_1|} \xi_{ai_\nu}^{m'} t_{i_\nu} + \sum_{\nu=|\theta_1|+1}^{|\theta\setminus\theta_1|} \xi_{ai_\nu}^{m'} t'_{i_\nu} \le \tau'_a,\ a \in A_\eta,\ \tau'_a \in \Im_\eta \quad (8)$$

где смеси Ξ_a в алгоритмах $a \in A_\eta$, в принципе сохранились, но в отношении (7) как самая продолжительная реализация алгоритма может оказаться другая — $\Xi_a^{m'}$, обусловленная сменой доминант. Соответственно изменится время $\tau'_a > \tau_a$, представляемое для решения в цикле УК Тц из расчета

$$\sum_{\chi=1}^{\chi_{n\eta}^{max}} \tau'_{n\chi} \le T_n,\ n = \overline{0,15},\ \tau'_{n\chi} \in \Im_\eta \quad (9)$$

Схему адаптации к текущему ф-состоянию на уровне аппаратного базиса архитектуры УК можно представить следующим образом

$$(\theta, T, A) \xrightarrow{\text{ф–отказ}} (\theta_1, T_1, -) \xrightarrow{\text{ф–адаптация}} \rightarrow$$

$$\rightarrow (\theta, T_1 \cup T'(7)) \xrightarrow{\text{А–адаптация}} (\theta, T_1 \cup T'(7), A_\eta(8)) \quad (10)$$

Факт восстановления готовности УК устанавливается самопроверкой, принципиально не отличающейся от общепринятой, за исключением ее повторов при необходимости и возможности принятия новых структурных решений в аппаратном базисе при первом же отри-

цательном результате очередного проверочного теста. После восстановления системы команд на основе качества отработки контрольных смесей выбирается наилучший вариант использования УК A_η.

Микропрограммный уровень архитектуры УК

Если отказ УК компенсируется на микропрограммном уровне его архитектуры организацией эмуляционных процессов для каждой отказавшей микрооперации в виде композиции, построенной на базе функционально полного ф-состояния:

$$(\forall \mu \in M \setminus M_1)(\exists \ell_\mu \in (M_1)^*)p(\ell_\mu \sim \mu), \quad (11)$$

то неизбыточным становится ухудшение быстродействия системы микрокоманд

$$(\forall \mu \in M \setminus M_1)p(t_{\ell_\mu} \sim t_\mu), \quad (12)$$

откуда вытекает снижение быстродействия системы команд.

Действительно, если время выполнения команд из $\theta 1$, не содержащих эмулируемые макрокоманды, сохраняется в рамках значений из T

$$(\forall \vartheta \in \theta_1)p(t_\vartheta = \sum_{\nu=1}^{|M|} \xi_{j_\nu}^{\vartheta} t_{Mj_\nu} = \sum_{\nu=1}^{|M_1|} \xi_{j_\nu}^{\vartheta} t_{Mj_\nu}), \quad (13)$$

т.к.

$$(\forall \mu_{j_\nu} \in M \setminus M_1)p(\xi_{j_\nu}^{\vartheta} = 0), \quad (14)$$

то для остальных $\vartheta \in \theta \setminus \theta_1$

$$t_\vartheta = \sum_{\varpi}^{|M_1|} \xi_{j_\omega}^{\vartheta} t_{Mj_\omega} + \sum_{\nu=|M_1|+1}^{|M|} \xi_{j_\nu}^{\vartheta} t'_{Mj_\nu}, \vartheta \in \theta \setminus \theta_1, \quad (15)$$

где

$$t'_{Mj_\nu} = \sum_{\varpi=1}^{|M_1|} \xi_{j_\omega}^{j_\nu} t_{Mj_\varpi}, \quad (16)$$

т.е.

$$t'_\vartheta = \sum_{\omega=1}^{|M_1|} \xi_{j_\omega}^{\vartheta} t_{Mj_\omega} + \sum_{\nu=|M_1|+1}^{|M|} \xi_{j_\nu}^{\vartheta} \sum_{\omega=1}^{|M_1|} \xi_{j_\omega}^{j_\nu} t_{Mj_\omega} =$$
$$= \sum_{\omega=1}^{|M_1|} \xi_{j_\omega}^{\vartheta} t_{Mj_\omega} + \sum_{\nu=|M_1|+1}^{|M|} \sum_{\omega=1}^{|M_1|} \xi_{j_\nu}^{\vartheta} \xi_{j_\omega}^{j_\nu} t_{Mj_\omega} =$$
$$= \sum_{\omega=1}^{|M_1|} \xi_{j_\omega}^{\vartheta} t_{Mj_\omega} + \sum_{\omega=1}^{|M_1|} \sum_{\nu=|M_1|+1}^{|M|} \xi_{j_\nu}^{\vartheta} \xi_{j_\omega}^{j_\nu} t_{Mj_\omega} =$$
$$= \sum_{\omega=1}^{|M_1|} (\xi_{j_\omega}^{\vartheta} + \sum_{\nu=|M_1|+1}^{|M|} \xi_{j_\nu}^{\vartheta} \xi_{j_\omega}^{j_\nu}) t_{Mj_\omega} \quad (17)$$

Снижение быстродействия системы команд приводит к уменьшению производительности УК и необходимости выбора наилучшего варианта управления объектом в рамках выражений (8) и (9).

Схема адаптации к текущему ф-состоянию на микрокомандном уровне архитектуры УК имеет вид:

$$(\theta, T, A) \xrightarrow{\text{ф–диагностироване}} (\theta_1, T_1, -) \xrightarrow{\text{ф–адаптация}}$$
$$\rightarrow (\theta, T_1 \cup T'(17), -) \xrightarrow{\text{А–адаптация}} (\theta, T_1 \cup T'(17), A_\eta(8)). \quad (18)$$

Обнаружение ф-отказа и факт восстановления готовности УК устанавливается традиционной самопроверкой, а функциональное диагностирование и организация адекватных эмуляционных процессов достигается как в предыдущем случае в соответствии методологией, опирающейся на процедуру адаптации, реализацию, в данном случае, в микропрограммной среде. Без применения дополнительного диагностического оборудования данный подход сталкивается с проблемой обеспечения достоверности результата методом самоконтроля неисправного устройства. А-адаптация осуществляется аналогично варианту, рассмотренному в подразделе ранее.

Командный уровень архитектуры УК

Организация эмуляционных процессов на данном уровне архитектуры УК [2–4] происходит путем замены отказавших команд эквивалентными композициями, построенными на базе сложившегося функционально полного ф-состояния $\theta 1$:

$$(\forall \vartheta \in \theta \setminus \theta_1)(\exists \ell_\vartheta \in (\theta_1)^*)p(\ell_\vartheta \sim \vartheta). \quad (19)$$

При этом очевидно, что быстродействие восстанавливаемой в логическом смысле системы команд уменьшается:

$$(\forall \vartheta \in \theta \setminus \theta_1)p(t_{\ell_\vartheta} > t_\vartheta), \quad (20)$$

откуда вытекает снижение производительности системы команд. Так, для произвольного алгоритма $a \in A$

$$\tau'_a \geq \sum_{\omega=1}^{|\theta_1|} \xi_{ai_\omega}^{m'} t_{i_\omega} + \sum_{\nu=|\theta_1|+1}^{|\theta|} \xi_{ai_\nu}^{m'} t'_{i_\nu} > \tau_a \quad (21)$$

где

$$t'_{i_\nu} = \sum_{\omega=1}^{|\theta_1|} \xi_{i_\omega}^{i_\nu} t_{i_\omega}, \vartheta_{i_\nu} \in \theta \setminus \theta_1, \vartheta_{i_\omega} \in \theta_1. \quad (22)$$

Таблица 1. Сводная таблица схем адаптации УК

№ п/п		ф-диагност.	ф-адаптация	А-адаптация
1	Логический уровень аппаратного базиса	θ_1 T_1 -	θ $T_1 \cup T'(7)$ -	θ $T_1 \cup T'(7)$ $A_\eta(8)$
2	Микропрограм-мный уровень архитектуры	$\theta_1(M_1)$ $T_1(T_{M_1})$ -	θ $T_1 \cup T'(7)$ -	θ $T_1 \cup T'(7)$ $A_\eta(8)$
3	Командный уровень архитектуры	θ_1 T_1 -	θ_1 $T_1 \cup T'(22)$	θ_1 $T_1 \cup T'(22)$ $A_\eta(23)$
4	Макрпрограм-мный уровень архитектуры	θ_1 T_1 -	θ_1 T_1 $\mu_1 \cup \mu'$	θ_1 T_1 $A_\eta(26)$

Тогда, аналогично выражения (17)

$$\tau'_a \geq \sum_{\omega=1}^{|\theta_1|} \xi^{m'}_{ai_\omega} t_{i_\omega} + \sum_{\nu=|\theta_1|+1}^{|\theta|} \xi^{m'}_{ai_\nu} \sum_{\omega=1}^{|\theta_1|} \xi^{i_\nu}_{i_\omega} t_{i_\omega} =$$

$$= \sum_{\omega=1}^{|\theta_1|} (\xi^{m'}_{ai_\omega} + \sum_{\nu=|\theta_1|+1}^{|\theta|} \xi^{m'}_{ai_\nu} \xi^{i_\nu}_{i_\omega}) t_{i_\omega}, \quad (23)$$

и А-адаптация завершается в соответствии с выражением (8).

Схема адаптации к текущему ф-состоянию на командном уровне архитектуры:

$$(\theta, T, A) \xrightarrow{\text{ф–диагностирование}} \theta_1, T_1, – \xrightarrow{\text{ф–адаптация}}$$

$$\rightarrow (\theta_1, T_1 \cup T'(22), –) \xrightarrow{\text{А–адаптация}} (\theta_1, T_1 \cup T'(22), A_\eta(23)). \quad (24)$$

В данном случае на самопроверку УК кроме определения вида технического состояния возлагаются и задачи функционального диагностирования с параллельным установлением эмуляционных процессов. А-адаптация осуществляется аналогично уровню архитектуры.

Макропрограммный уровень архитектуры УК

Эмуляционные процессы на этом уровне архитектуры УК [1] строятся на основе замены нетерминальных макрокоманд из состава M\M1, где M1 — множество работоспособных, опирающихся на сохранившуюся подсистему команд $\theta 1$, эквивалентными макрокомандами из некоторого дополнительного (запасного) списка M':

$$(\forall M \in \mu \setminus \mu_1, M \in (\theta_1)^*)(\exists M' \in \mu', M' \in (\theta_1)^*,$$

$$|\mu_1| + |\mu'| = |\mu| p(M' \sim M). \quad (25)$$

В результате восстановления работоспособности УК без восстановления исходной системы команд θ быстродействие оставшихся исправными команд T1 сохраняется, но меняются смеси рабочих алгоритмов, а, следовательно, производительность УК, по схеме

$$\tau'_a \geq \sum_{\nu=1}^{|\mu_1|} \xi^{m'}_{ак_\nu} t_{Мк} + \sum_{\nu=|\mu_1|+1}^{|\mu|} \xi^{m'}_{ак_\nu} t'_{Мк} =$$

$$= \sum_{\nu=1}^{|\mu_1|} \xi^{m'}_{ак_\nu} \sum_{\omega=1}^{|\theta_1|} \xi_{Mi_\omega} t_{i_\omega} + \sum_{\nu=|\mu_1|+1}^{|\mu|} \xi^{m'}_{ак_\nu} \sum_{\omega=1}^{|\theta_1|} \xi'_{Mi_\omega} t_{i_\omega} =$$

$$= \sum_{\omega=1}^{|\theta_1|} (\sum_{\nu=1}^{|\mu_1|} \xi^{m'}_{ак_\nu} \xi_{Mi_\omega} + \sum_{\nu=|\mu_1|+1}^{|\mu|} \xi^{m'}_{ак_\nu} \xi'_{Mi_\omega}) t_{i_\omega} \quad (26)$$

А-адаптация завершается в соответствии с выражением (26) и (9).

Схема адаптации к текущему ф-состоянию на макрокомандном уровне архитектуры примет вид:

$$(\theta, T, A) \xrightarrow{\text{ф–диагностирование}} (\theta_1, T_1, –) \xrightarrow{\text{ф–адаптация}}$$

$$\rightarrow (\theta_1, T_1, \mu_1 \cup \mu') \xrightarrow{\text{А–адаптация}} (\theta_1, T_1, A_\eta(26)). \quad (27)$$

Требования к самопроверке УК аналогичны предыдущему уровню архитектуры.

Заключение

Обобщение результатов сравнительного анализа целесообразно выполнять, опираясь на справочную таблицу схем адаптации УК к текущему функциональному состоянию (таблица 1).

По мнению автора, итоги анализа могут быть сформулированы следующим образом:

1. Эмуляционные процессы на любом уровне архитектуры УК тем или иным образом проявляются на базовом [2–4] командном уровне, в конечном счете характеризующем текущее ф-состояние (состав команд) и ф-состояние после адаптации (быстродействие и остаточная производительность). В связи с этим методику адаптации структурно-устойчивого УК к текущему ф-состоянию целесообразно строить по принципу «ядра и оболочки»: в ее основу положить алгоритмы адаптации на командном уровне архитектуры («ядро»), которые, по мере необходимости модернизируются под особенности иного функционального уровня, создавая, своего рода, «оболочку».
2. Основным средством функционального диагностирования в условиях непривлечения специальной диагностирующей аппаратуры остается предэтапная самопроверка, которая может совмещать и задачи функциональной адаптации. Это обстоятельство острее, чем обычно, ставит проблему достоверности результатов, полученных по данным функционирования неисправного УК.
3. Не смотря на то, что к настоящему моменту проблема комплексирования эмуляционных процессов на различных уровнях архитектуры УК решена лишь в методологическом плане, необходимо исследовать возможные пути комплексирования и соответствующих процессов адаптации УК к текущему ф-состоянию в широком смысле.
4. Для математической постановки научной задачи и состава частных задач исследования необходимо разработать математическую модель процесса адаптации структурно-устойчивого УК к текущему ф-состоянию.

ЛИТЕРАТУРА

1. Беляков А.Ю., Харитонов В. А. и др. Формальная система в задаче синтеза средств организации эмуляционных процессов в управляющих вычислительных системах с адаптивными интерпретаторами. Сборник научных трудов ГосНИИУМС. Вып. 47.— Пермь, 1998. С. 131–141.
2. Зарубский В. Г. Вопросы разработки перспективных интегрированных систем охраны, отвечающих требованиям повышенной живучести, на базе структурно-устойчивых управляющих компьютеров. Вестник Пермского института ФСИН России. № 1 (5)/ 2012. С 4–9.
3. Зарубский В. Г. Особенности организации процесса функционального диагностирования управляющего компьютера повышенной живучести. Надежность. № 3/2016.— С. 35–38.
4. Зарубский В.Г., Рыбаков А. П. Математическая модель процесса адаптации управляющего компьютера интегрированной системы охраны к текущему функциональному состоянию. Вестник Воронежского института МВД России. № 1/ 2012. С. 170–178.
5. Олейников А.В., Харитонов В. А. Отказоустойчивая трансляция в динамических интерпретаторах управляющих программных комплексов. Моделирование вычислительных систем: межвузовский сборник научных трудов.— Пермь: ПГУ, 1991. С. 126–132.
6. Тюрин С. Ф. Скользящее резервирование толерантных элементов. Надежность. 2017. Т. 17. № 1 (60). С. 17–21.
7. Тюрин С. Ф. Функционально полные толерантные элементы ПЛИС FPGA для аэрокосмических вычислительных комплексов. Вестник Сибирского государственного аэрокосмического университета им. академика М. Ф. Решетнева. 2016. Т. 17. № 2. С. 490–497.
8. Харитонов В. А. Основы теории живучести функционально-избыточных систем. С.-Пб.: СПИИРАН, 1993.— 60 с.

Журнал «Современная наука: актуальные проблемы теории и практики»